PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-132531
(43)Date of publication of application : 12.05.2000

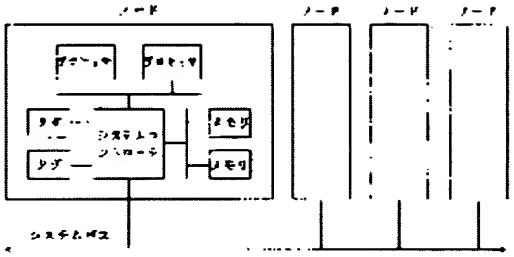
(51)Int.Cl. G06F 15/177
G06F 15/16

(21)Application number : 10-301948
(22)Date of filing : 23.10.1998

(71)Applicant : PFU LTD
(72)Inventor : HIKONO ATSUSHI
TAKEBE KENJI
SATO MASAMI

(54) MULTIPROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To make effectively performable a process to keep the coherency of caches in a multiprocessor system which includes plural processors having the caches.
SOLUTION: When a system controller receives a memory access request from its follower processor, the controller broadcasts the received access request to a system bus and also outputs the cache state included in the tag information corresponding to a memory space x to be accessed to the system bus. On receiving the broadcast memory access request, the system controller of another node outputs the cache state included in the tag information corresponding to the space x to the system bus, gives a request to its follower processor or memory for a necessary process based on the state of the cache existing on the system bus and returns a response to the system controller of a master after the requested process is over.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of

rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－132531
（P2000－132531A）
(43)公開日　平成12年５月12日(2000.5.12)

(51)Int.Cl.'		識別記号	FI			テーマコード'(参考)
Ｇ０６Ｆ	15/177	６８２	Ｇ０６Ｆ	15/177	６８２Ｊ	５Ｂ０４５
		６７４			６７４Ａ	
	15/16	６４５		15/16	６４５	

審査請求　未請求　請求項の数５　ＯＬ　（全 16 頁）

(21)出願番号　特願平10－301948

(22)出願日　平成10年10月23日(1998.10.23)

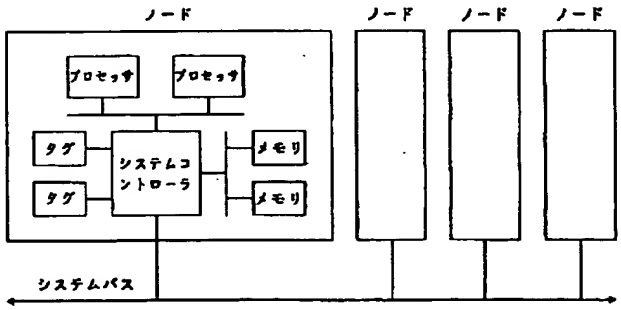
(71)出願人　000136136
株式会社ピーエフユー
石川県河北郡宇ノ気町字宇野気ヌ98番地の２
(72)発明者　彦野　厚志
石川県河北郡宇ノ気町字宇野気ヌ98番地の２　株式会社ピーエフユー内
(72)発明者　武部　建治
石川県河北郡宇ノ気町字宇野気ヌ98番地の２　株式会社ピーエフユー内
(74)代理人　100080894
弁理士　京谷　四郎　（外１名）

最終頁に続く

(54)【発明の名称】　マルチプロセッサ

(57)【要約】
【課題】　キャッシュを持つ複数のプロセッサを有するマルチプロセッサにおいて、キャッシュのコヒーレンシーを維持するための処理を効率良く行うこと。
【解決手段】　システム・コントローラが配下のプロセッサからメモリ・アクセス要求を受け取ると、当該メモリ・アクセス要求をシステム・バス上にブロードキャストすると共に、アクセス対象のメモリ空間ｘに対応するタグ情報が持つキャッシュ状態をシステム・バス上に出力する。ブロードキャストされたメモリ・アクセス要求を受け取った他のノードのシステム・コントローラは、メモリ空間ｘに対応するタグ情報が持つキャッシュ状態をシステム・バス上に出力し、次いでシステム・バス上のキャッシュの状態を参照して配下のプロセッサ又はメモリに対して必要な処理を要求し、要求した処理が終了した後にマスタのシステム・コントローラに対して応答を返す。

本発明の原理説明図



1

【特許請求の範囲】

【請求項１】　キャッシュを持つプロセッサ，システム・コントローラおよび主メモリの一部を構成するメモリを有する複数個のノード、並びに複数個のノードを接続するシステム・バスを具備するマルチプロセッサにおいて、

各システム・コントローラは、配下のプロセッサからメモリ・アクセス要求を受けた時に、当該メモリ・アクセス要求をシステム・バスを介して他の全てのシステム・コントローラにブロードキャストし、

配下のプロセッサ又は他のシステム・コントローラからのメモリ・アクセス要求を受けた時に、自己のキャッシュのタグ部の中に当該メモリ・アクセスの対象となるメモリ空間に対応するタグ情報が存在するか否かを調べ、存在する場合には該当するタグ情報が持つキャッシュ状態をシステム・バスに出力し、存在しない場合には無効を示すキャッシュ状態をシステム・バスに出力し、

システム・バス上のキャッシュ状態を参照し、配下のプロセッサ又はメモリを制御する必要がある場合には配下のプロセッサ又はメモリを制御することを特徴とするマルチプロセッサ。

【請求項２】　ブロードキャストされたメモリ・アクセス要求を受け取ったシステム・コントローラは、システム・バス上のキャッシュ状態を参照して独自に配下のプロセッサに対する処理が必要か否かを判断し、必要と判断した場合には配下のプロセッサに必要とされる処理を要求することを特徴とする請求項１のマルチプロセッサ。

【請求項３】　自己の配下に存在する複数個のプロセッサに対してインバリデート，コピーバック又はコピーバック・インバリデートの処理要求を発行したシステム・コントローラは、上記複数個の処理要求に対する処理終了の応答を全て受け取った段階で、メモリ・アクセス要求をブロードキャストしたシステム・コントローラに対して纏めて処理終了の応答を返すことを特徴とする請求項２のマルチプロセッサ。

【請求項４】　メモリ・アクセスの対象となるメモリ空間を持つノードのシステム・コントローラは、メモリ・アクセス要求を受け取った時に自己のメモリに対するアクセスを開始し、次いでシステム・バス上のキャッシュ状態を参照してメモリ・アクセスが必要か否かを調べ、必要ないと判断した場合には当該メモリ・アクセスをキャンセルすることを特徴とする請求項１のマルチプロセッサ。

【請求項５】　メモリ・アクセス要求をシステム・バスにブロードキャストしたシステム・コントローラは、システム・バス上のキャッシュ状態を参照して、プロセッサへの処理が必要なシステム・コントローラ又はリソース・メモリを制御するシステム・コントローラからのみ応答を待ち、関係のないシステム・コントローラからは

2

応答を待たないことを特徴とする請求項１，請求項２，請求項３または請求項４マルチプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のプロセッサのキャッシュのコヒーレンシーの制御を複数のシステム・コントローラがシステム・バスを介して行うマルチプロセッサに関するものである。

【０００２】

【従来の技術】従来、複数のプロセッサを制御するシステム・コントローラを複数個持つマルチプロセッサにおいては、プロセッサ又はバス・ブリッジからメモリ・アクセス要求を受けたシステム・コントローラ（マスタのシステム・コントローラ）が他のプロセッサを制御しているシステム・コントローラ（スレーブのシステム・コントローラ）から各プロセッサのキャッシュの状態を読み出し、必要に応じて他のシステム・コントローラが制御するプロセッサが持つキャッシュに対するコピーバック要求やインバリデート要求を当該システム・コントローラに対してシステム・バスを経由して送っていた。

【０００３】

【発明が解決しようとする課題】従来のシステムでは、キャッシュのコヒーレンシー（ｃｏｈｅｒｅｎｃｙ）を保つために、或る一つのメモリ・アクセス要求に対して複数のバス・トランザクションの遣り取りがシステム・バス上で必要であった。例えば、或るメモリ・アクセスに対して複数のプロセッサのキャッシュのインバリデート（ｉｎｖａｌｉｄａｔｅ）作業が必要な場合は、その数だけメモリ・アクセス要求を受けたマスタのシステム・コントローラがシステム・バスを介してインバリデート作業が必要なプロセッサを配下に持つシステム・コントローラに対してインバリデート要求を送らなければならなかった。

【０００４】そのため、幾つものバス・トランザクションがシステム・バスに出てしまい、システム・バスを占有してしまっていた。そのため、プロセッサのキャッシュに対するインバリデート要求やコピーバック要求が遅れてしまうと共に、他のアクセスと競合する確率が高くなり、システム全体の性能の低下を招いていた。本発明は、この点に鑑みて創作されたものであって、複数のプロセッサの各々がキャッシュを有するマルチプロセッサにおいて、キャッシュのコヒーレンシーを維持するための処理を効率よく行い得るようにすることを目的としている。

【０００５】

【課題を解決するための手段】図１は本発明の原理説明図である。本発明では、キャッシュの状態を複数のシステム・コントローラで共有する。或るメモリ・アクセス要求が発生したら、当該メモリ・アクセス要求を受けたマスタのシステム・コントローラは、システム・バスに

当該アクセス要求をブロードキャストする。システム・バスに接続されている各システム・コントローラは、当該ブロードキャストを受信すると、自分が制御するプロセッサのキャッシュの状態をシステム・バス上に出力する。各システム・コントローラは、システム・バス上のキャッシュの状態を見ることで、配下のプロセッサに対して、コピーバック作業やインバリデート作業が必要か否かを独自に判断することが出来る。

【０００６】システム・コントローラは、メモリ・アクセス要求を受け取ると、マスタのシステム・コントローラから要求されることなく、必要に応じてコピーバック作業やインバリデート作業を即座に開始する。

【０００７】また、ブロードキャストされたメモリ・アクセス要求がリードの場合には、リソース・メモリを持つシステム・コントローラは、そのメモリへのリード・アクセスを開始し、次いで各システム・コントローラから出力されたキャッシュの状態が最新データがキャッシュに存在することを示しているか否かを調べ、最新データがキャッシュに存在する場合には当該メモリ・アクセスをキャンセルする。なお、リソース・メモリとは、ブロードキャストされたメモリ・アクセス要求の対象となるメモリ空間を持つメモリを意味している。

【０００８】処理が必要と判断したシステム・コントローラは、自分が制御するプロセッサやメモリへの処理が全て終了した段階で、マスタのシステム・コントローラに応答を返す。マスタのシステム・コントローラは、応答が必要なシステム・コントローラからの応答を待つと共に、自分が制御するプロセッサへの処理が必要な場合はその処理を行った後、メモリ・アクセスを終結させる。

【０００９】本発明によれば、システム・コントローラの各々が独自にキャッシュのコヒーレンシーを維持するための作業を行うために、システム・バスの使用を最低限にすることができ、性能の低下を防げる。また、各々のシステム・コントローラが独自に判断するために、いちはやくプロセッサに対するコピーバック作業やインバリデート作業，メモリ・アクセス，メモリ・アクセスのキャンセルの開始が可能である。

【００１０】

【発明の実施の形態】図２は本発明のマルチプロセッサの構成例を示す図である。同図において、１００はノード、１１１と１１２はプロセッサ、１２１と１２２はタグ部、１３０はシステム・コントローラ、１４０はクロスバ・スイッチ、１５１と１５２はメモリ、１６０はバス・ブリッジ、２００はノード、２１１と２１２はプロセッサ、２２１と２２２はタグ部、２３０はシステム・コントローラ、２４０はクロスバ・スイッチ、２５１と２５２はメモリ、２６０はバス・ブリッジ、３００はノード、４００もノードをそれぞれ示している。

【００１１】ノード１００，２００，３００，４００は同じ構成を持っているので、主としてノード１００について説明する。バス・ブリッジ１６０の先には入出力装置が接続されている。プロセッサ１１１は内部にキャッシュ（図示せず）を有しており、キャッシュはデータ部とタグ部とから構成されている。プロセッサ１１２も内部にキャッシュを有している。タグ部１２１の内容はプロセッサ１１１のキャッシュのタグ部の内容と実質的に同じであり、タグ部１２２の内容はプロセッサ１１２のキャッシュのタグ部の内容と実質的に同じである。

【００１２】システム・コントローラ１３０は、アドレスを一括制御するものである。クロスバ・スイッチ１４０は、システム・コントローラ１３０の指示に従って、データの流れを制御するものである。例えば、バス・ブリッジ１６０のデータ・バスをシステム・バスに含まれるデータ・バスに接続したり、プロセッサ１１１のデータ・バスをメモリ１５１のデータ・バスに接続したりする。メモリ１５１は主記憶の一部を構成するものである。メモリ１５２も同様である。例えば、メモリ１５１には主記憶アドレスのビット６～８が０００であるデータが格納され、メモリ１５２には主記憶アドレスのビット６～８が００１のデータが格納される。また、ノード２００のメモリ２５１には主記憶アドレスのビット６～８が０１０のデータが格納され、ノード２００のメモリ２５２には主記憶アドレスのビット６～８が０１１のデータが格納される。

【００１３】タグ部１２１はN個（Nは２，３，…）のタグ情報を記憶することが出来る。タグ情報は、アドレスとキャッシュの状態を持つ。キャッシュの状態種別としては、M，O，S，Iがある。MはＥｘｃｌｕｓｉｖｅ＆ＰｏｔｅｎｔｉａｌｙＭｏｄｉｆｙを示し、OはＳｈａｒｅｄ－Ｍｏｄｉｆｙを示し、SはＳｈａｒｅ－Ｃｌｅａｎを示し、IはＩｎｖａｌｉｄａｔｅを示す。

【００１４】タグ部１２１の中に存在する或るタグ情報のキャッシュの状態がMであると言うことは、当該タグ情報に対応する主記憶上のメモリ空間がプロセッサ１１１によって占有され、且つ当該タグ情報に対応するプロセッサ１１１のキャッシュのデータ・ブロックが書き換えられている可能性があることを示す。タグ部１２１の中に存在する或るタグ情報のキャッシュの状態がOであると言うことは、当該タグ情報に対応する主記憶上のメモリ空間がプロセッサ１１１を含む複数のプロセッサによって共有され、且つ当該タグ情報に対応するプロセッサ１１１のキャッシュのデータ・ブロックが書き換えられている可能性があることを示す。

【００１５】タグ部１２１の中に存在する或るタグ情報のキャッシュの状態がSであると言うことは、当該タグ情報に対応する主記憶上のメモリ空間がプロセッサ１１１を含む複数のプロセッサによって共有され、且つ当該タグ情報に対応するプロセッサ１１１のキャッシュのデータ・ブロックが書き換えたものを共有している場合も

あることを示す。タグ部１２１の中に存在する或るタグ情報のキャッシュの状態がＩであると言うことは、当該タグ情報に対応するプロセッサ１１１のキャッシュのデータ・ブロックが無効であることを示す。

【００１６】ノード１００，２００，３００，４００は、システム・バスによって接続されている。システム・バスは、アドレス・バス，データ・バス，リプライ・バス，スヌープ（ｓｎｏｏｐ）バスから構成されている。各ノードのシステム・コントローラはアドレス・バス，リプライ・バス，スヌープ・バスに接続され、クロスバ・スイッチはデータ・バスに接続されている。

【００１７】アドレス・バスには、以下に示すような複数の情報が入る。

○アドレス

○タイプ（アクセスの種類を示す。ＲＤＳ，ＲＤＳＡ，…など）

○マスタＩＤ（アクセス発生元。例えばＣＰＵのＩＤなど）

○クラス（処理の優先順位）

【００１８】スヌープ・バスはシステム・コントローラから４本出ている。１個のプロセッサあたり２本で、４つのキャッシュの状態を示す。図示のように、４個のシステム・コントローラが存在し、各システム・コントローラの配下に２個のプロセッサが存在する場合には、スヌープ・バスは１６本の信号線で構成される。

【００１９】リプライ・バスは複数本の信号線で構成される。以下に構成要素を示す。

○マスタＩＤ（どのマスタＩＤのアクセスに介するリプライかを示す。）

○クラス（どのクラスに対するリプライかを示す。）

○タイプ（リプライの種類を示す。ＲＢＳ，ＳＷＢ，ＰＳＡＣＫなど）

ＲＢＳとは、Ｒｅａｄ　Ｂｌｏｃｋ　Ｓｈａｒｅｄの略であり、メモリ・リードに対するリプライである。ＳＷＢとは、Ｓｌａｖｅ　Ｗｒｉｔｅ　Ｂｌｏｃｋの略であり、メモリ・ライトでデータをスレーブに送るリプライである。ＰＳＡＣＫとは、インバリデート終了をマスタのシステム・コントローラに送るリプライである。

【００２０】プロセッサが出すメモリ・アクセス要求には、以下に示すような４種類のリード要求と，２種類のライト要求とが存在する。

ＲＤＳ　Ｒｅａｄ　Ｔｏ　Ｓｈａｒｅ
ＲＤＳＡ　Ｒｅａｄ　Ｔｏ　Ｓｈａｒｅ　Ａｌｗａｙｓ
ＲＤＤ　Ｒｅａｄ　Ｔｏ　Ｄｉｓｃａｒｄ
ＲＤＯ　Ｒｅａｄ　Ｔｏ　Ｏｗｎ
ＷＲＢ　Ｗｒｉｔｅ　Ｂａｃｋ
ＷＲＩ　Ｗｒｉｔｅ　Ｉｎｖａｌｉｄａｔｅ

【００２１】図３はシステム・コントローラの構成例を示す図である。同図において、１３１はプロセッサ・バス制御部、１３２はアドレス・バス制御部、１３３はスヌープ・バス制御部、１３４はクロスバ制御部、１３５はメモリ制御部をそれぞれ示している。

【００２２】プロセッサ・バスとは、プロセッサに接続されているアドレス・バスのことである。システム・コントローラ１３０は、プロセッサ・バス制御部１３１やアドレス・バス制御部１３２，スヌープ・バス制御部１３３，クロスバ制御部１３４，メモリ制御部１３５を有している。他のシステム・コントローラも同様な構成を有している。

【００２３】プロセッサ・バス制御部１３１は、プロセッサ・バスからデータを取り込んだり、プロセッサ・バスにデータを出力したりする。アドレス・バス制御部１３２は、システム・バスに含まれるアドレス・バスにデータを出力したり、これからデータを取り込んだりする。プロセッサ・バス制御部１３１とアドレス・バス制御部１３２の間には、データを遣り取りするための信号線が設けられている。

【００２４】スヌープ・バス制御部１３３は、スヌープ・バスにデータを出力したり、スヌープ・バスからデータを取り込んだりすると共に、タグ部１２１の制御を行う。なお、図にはタグ部１２１しか示されていないが、タグ部１２２（図２を参照）もスヌープ・バス制御部１３３によって制御される。クロスバ制御部１３４は、クロスバ・スイッチやリプライ・バス，データ・バスの制御を行う。プロセッサ・バス制御部１３１とクロスバ制御部１３４の間にはデータを遣り取りするための信号線が設けられ、アドレス・バス制御部１３２とクロスバ制御部１３４の間にもデータを遣り取りするための信号線が設けられている。

【００２５】メモリ制御部１３５は、メモリの制御を行う。クロスバ制御部１３４とメモリ制御部１３５の間の信号線はデータの転送に使用される。また、スヌープ・バス制御部１３３からメモリ制御部１３５には制御信号が送られる。

【００２６】図４は本発明の動作を説明する図である。同図において、３１１と３１２はプロセッサ、３２１と３２２はタグ部、３３０はシステム・コントローラ、３４０はクロスバ・スイッチ、３６０はバス・ブリッジをそれぞれ示す。　～　は処理の順序を示す。なお、図２と同一符号は同一物を示す。また、ノード２００，３００の中にはメモリ１５１，１５２に相当するものが存在するが、図示されていない。

【００２７】図４に示すように、プロセッサ１１１が或るメモリ空間ｘに所有権つきのリード（キャッシュを書き換え可能な状態にするリード）をシステム・コントローラ１３０に対して要求したと仮定する。なお、所有権付きリードとは、前述のＲＤＯのことである。システム・コントローラ１３０は、当該メモリ・リード要求をアドレス・バスを介してブロード・キャストする。なお、ブロード・キャストとは、計算機システムが複数のシス

テム・コントローラを持つ場合、一つのプロセッサが出してきたメモリ・アクセス要求を他のシステム・コントローラの全てに見せることを意味している。

【００２８】システム・コントローラ１３０は、メモリ・リード要求をブロードキャストした後に、タグ部１２１，１２２を参照して当該メモリ・リードの対象となるメモリ空間ｘに対応するキャッシュの状態を求め、求めたキャッシュの状態をスヌープ・バス上に出力する。なお、該当するタグ情報がタグ部に存在しない場合は、インバリッドがスヌープ・バス上に出力される。

【００２９】システム・コントローラ２３０は、ブロード・キャストされたメモリ・アクセス要求を受信すると、タグ部２２１，２２２を参照してメモリ空間ｘに対応するキャッシュの状態を求め、求めたキャッシュの状態をスヌープ・バス上に出力する。同様に、システム・コントローラ３３０は、ブロード・キャストされたメモリ・アクセス要求を受信すると、タグ部３２１，３２２を参照してメモリ空間ｘに対応するキャッシュの状態を求め、求めたキャッシュの状態をスヌープ・バス上に出力する。

【００３０】各システム・コントローラはスヌープ・バス上に出力されたキャッシュの状態を認識し、即座に必要な処理を開始する。例えば、スヌープ・バス上に出力されたプロセッサ２１１のキャッシュの状態がＳｈａｒｅｄ－Ｍｏｄｉｆｙ（キャッシュに最新のデータが存在する状態）、スヌープ・バス上に出力されたプロセッサ３１１のキャッシュの状態がＳｈａｒｅ－Ｃｌｅａｎ（メモリ又は他のキャッシュの写しを共有している状態）、スヌープ・バス上に出力された他のキャッシュの状態がＩｎｖａｌｉｄ（無効）であったと仮定する。

【００３１】システム・コントローラ２３０は、システム・コントローラ１３０によって所有権付きメモリ・リードが要求され且つ当該メモリ・リードの対象となるメモリ空間ｘに対応するタグ情報のキャッシュの状態がＳｈａｒｅｄ－Ｍｏｄｉｆｙであるので、プロセッサ２１１に対してコピーバック・インバリデート要求を発行する。

【００３２】このコピーバック・インバリデート要求を受け取ると、プロセッサ２１１は、キャッシュのデータ部から該当するデータ・ブロックを読み出してシステム・コントローラ２３０に渡すと共に、キャッシュのタグ部に存在するメモリ空間ｘに対応するタグ情報のキャッシュの状態をＩｎｖｌｉｄにし、処理終了の応答をシステム・コントローラ２３０に返す。プロセッサ２１１からの処理終了の応答を受け取ると、システム・コントローラ２３０は、タグ部２２１の該当するタグ情報のキャッシュの状態をＩｎｖａｌｉｄにし、渡されたデータ・ブロックをシステム・バスに含まれるデータ・バスに出力し、マスタのシステム・コントローラ１３０にリプライ・バスを経由して応答を返す。

【００３３】システム・コントローラ３３０は、マスタのシステム・コントローラが所有権付きメモリ・リードを要求し且つプロセッサの３１１の該当するキャッシュの状態がＳｈａｒｅ－Ｃｌｅａｎであるので、プロセッサ３１１に対してインバリッド要求を発行する。このインバリッド要求を受け取ると、プロセッサ３１１は、キャッシュのタグ部の中に存在するメモリ空間ｘに対応するタグ情報のキャッシュの状態をＩｎｖａｌｉｄにし、処理終了の応答をシステム・コントローラ３３０に返す。この応答を受け取ると、システム・コントローラ３３０は、タグ部３２１の中に存在する該当するタグ情報のキャッシュの状態をＩｎｖａｌｉｄにし、マスタのシステム・コントローラ１３０にリプライ・バスを経由して処理終了の応答を返す。

【００３４】一方、システム・コントローラ１３０は、各システム・コントローラ２３０，３３０がキャッシュの状態をスヌープ・バス上に出力した所でスヌープ・バス上のキャッシュの状態を参照し、最新データがプロセッサ２１１のキャッシュにあると認識して、メモリ・リードを止めて、システム・コントローラ２３０からの応答およびシステム・コントローラ３３０からの応答を待ち、その後、プロセッサ１１１に対してリード・データを渡す。このリード・データを受け取ると、プロセッサ１１１は、リード・データをキャッシュのデータ部に格納すると共に、アドレスとＥｘｃｌｕｓｉｖｅ＆Ｐｏｔｅｎｔｉａｌｙ　Ｍｏｄｉｆｙと言うキャッシュの状態とを持つタグ情報をキャッシュのタグ部に書き込む。

【００３５】また、上記の例に加えて、例えばプロセッサ２１２のキャッシュのタグ部の中に存在するメモリ空間ｘに対応するタグ情報のキャッシュの状態がＩｎｖａｌｉｄ（無効）ではなくてＳｈａｒｅ－Ｃｌｅａｎの状態であると仮定すると、システム・コントローラ２３０はプロセッサ２１１に対してはコピーバック・インバリデートを発行し、プロセッサ２１２に対してはインバリデートを発行する。このような場合には、システム・コントローラ２３０はプロセッサ２１１，２１２からの応答を待ち合わせ、両方からの応答があった段階でシステム・コントローラ１３０に処理終了を告げる応答を返す。このようにすることにより、マスタのシステム・コントローラに対する応答が一度で済む。

【００３６】上述の説明はプロセッサ１１１から所有権つきのリードが発行された場合の動作を説明するものであるが、プロセッサ１１１から通常のリード要求（ＲＤＳ／ＲＤＳＡ）が発行された場合の動作を以下に説明する。プロセッサ１１１が通常のリード要求を発行すると、システム・コントローラ１３０はシステム・バス上に当該リード要求をブロード・キャストし、システム・コントローラ１３０，２３０，３３０はタグ部の内容を確認して、スヌープ・バスにタグのステート（キャッシュの状態を示す）を出力する。

【００３７】スヌープ・バス上のタグのステートが全てＩであるとすると、データはメモリから読み出される。何れのメモリからリードするかは、リード要求のアドレスによって決まる。

【００３８】スヌープ・バス上に出力されたプロセッサ１１２のタグのステートがＳであり、他のタグのステートがＩであったとすると共に、リード要求で指定されるアドレスがメモリ２５１の空間であったと仮定する。この場合、リード要求で指定された最新データは、メモリ２５１およびプロセッサ１１２のキャッシュに存在する。他のノード２００のメモリ２５１からノード１００にデータを転送するよりも自ノード１００内で終結する方が処理を早く終わらせることができ、システム・バスを使用しなくても済む。従って、プロセッサ１１２のキャッシュからデータを取ってくることを選択する。システム・コントローラ１３０は、プロセッサ１１２に対してコピーバックを発行し、この結果得られるデータをプロセッサ１１１にデータを転送する。リード要求で指定されるアドレスがメモリ１５１の空間であったと仮定する。この場合は、コピーバックよりメモリ・リードの方が速いので、システム・コントローラ１３０はメモリ１５１から該当するデータをリードし、リード・データをプロセッサ１１１に渡す。

【００３９】スヌープ・バス上のプロセッサ２１２のタグのステートがMであり、他のプロセッサのタグのステートがＩであったと仮定する。この場合は、最新データはプロセッサ２１２しか持っていないので、システム・コントローラ２３０がプロセッサ２１２にコピーバックを要求し、コピーバックの結果得られたデータをプロセッサ２１２からプロセッサ１１１に転送する。

【００４０】次に、プロセッサがライト・インバリデート（ＷＲＩ）を発行した場合の動作について説明する。プロセッサ１１１からシステム・コントローラ１３０にライト・インバリデート要求が出されると、システム・コントローラ１３０は当該要求をシステム・バス上にブロード・キャストし、各システム・コントローラ１３０，２３０，３３０はタグ部の内容を確認し、タグのステートをスヌープ・バス上に出す。このとき、各システム・コントローラは、スヌープ・バス上のタグのステートがＩ以外のＳ又はＯ又はMになっているプロセッサに対してインバリデート要求を出し、インバリデートが終了した段階でシステム・コントローラ１３０に応答を返す。システム・コントローラ１３０は全てのインバリデートが終了した段階で、メモリ・ライトを行い、処理を終結させる。

【００４１】図５はプロセッサ・バスからリード要求を受け取ったマスタのシステム・コントローラの動作を示す図である。同図において、ＰＢはプロセッサ・バス、ＴＷはシステム・バスをそれぞれ示す。

【００４２】システム・コントローラは、プロセッサ・バスからコヒーレント・リード（ＲＤＳ，ＲＤＳＡ，ＲＤＤまたはＲＤＯ）を受け取ると、処理中のメモリ・アクセス要求のアドレスと今回のリード要求のアドレスが一致しているか否かを調べる。両者が一致している場合は、今回のリード要求をブロッキング・バッファに格納する。両者が一致していない場合は、今回のリード要求をシステム・バスに出力すると共に、今回のリード要求で指定されたメモリ空間が自己のノードの中に存在するか否かを調べる。自己のノードの中に存在する場合には、メモリからデータをリードしてリード・データをクロスバ・スイッチのバッファに入れるための処理を開始する。

【００４３】リード要求をシステム・バスに出力した後、システム・コントローラはリード要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。そして、システム・コントローラは、読み込んだキャッシュの状態を参照して種々の動作を行う。

【００４４】或る場合には、「ＰＢ　ＳＲＥＱ　ＣＰＢ／Ｉ　ＩＮＶ」を発行する。即ち、プロセッサ・バスを経由してプロセッサにコピーバックやコピーバック・イバリデート，インバリデート要求を送る。そして、プロセッサ・バスを経由してリプライやＳＡＣＫか送られて来るのを待つ。「ＳＡＣＫ」とは、インバリデート終了のリプライである。

【００４５】或る場合には、「ＴＷ　Ｒｅｐｌｙ　ＳＡＣＫ」を待つ。即ち、システム・バスを経由してリプライやＳＡＣＫが送られてくるのを待つ。ある場合には、「ＴＷ　Ｒｅｐｌｙ　ＲＢＵ／Ｓ」を待つ。即ち、システム・バスを経由してＲＢＳ（リード・ブロック・シェア）又はＲＢＵ（リード・ブロック・アンシェア）のリプライが送られて来るのを待つ。ＲＢＵやＲＢＳは、メモリ・リードに対するリプライである。或る場合にはメモリ対するリード処理をキャンセルし、或る場合にはメモリ・リードが終了するのを待つ。

【００４６】待ち合わせ処理が終了した後、システム・コントローラは、ＲＢＵ又はＲＢＳ又はＯＡＫ（オーナシップ・アック）のリプライをマスタに送る。この場合のマスタとは、リード要求発行元のプロセッサである。また、必要に応じて「ＣＲＡＢ」をスレーブのプロセッサに送る。「ＣＲＡＢ」とは、コヒーレンシ・リード・アック・ブロックを意味している。

【００４７】プロセッサ側にデータを送った後、システム・コントローラは「ＴＷ　ＣＥＮＤ」を発行する。「ＴＷ　ＣＥＮＤ」は、システム・バス側の処理が終了したことを意味している。「ＴＷ　ＣＥＮＤ」が発行されると、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態が解除され、次いでタグの更新が行われる。

【００４８】図６はシステム・バスからリード要求を受

け取ったスレーブのシステム・コントローラの動作を示す図である。システム・コントローラは、システム・バスからコヒーレント・リード（ＲＤＳ，ＲＤＳＡ，ＲＤＤまたはＲＤＯ）を受信すると、受信したリード要求で指定されたメモリ空間が自己のノードの中に存在するか否かを調べる。自己のノードの中に存在する場合には、メモリからデータをリードしてリード・データをクロスバ・スイッチのバッファに入れるための処理を開始する。

【００４９】リード要求を受信した後、システム・コントローラはリード要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。そして、システム・コントローラは、読み込んだキャッシュの状態を参照して種々の動作を行う。

【００５０】或る場合には、「ＰＢ　ＣＰＢ／ＣＰＩ＋α（ＩＮＶ）」を発行する。即ち、或るプロセッサに対してはコピーバック，コピーバック・インバリデート又はインバリデートを発行し、必要に応じて他のプロセッサに対してもコピーバック，コピーバック・インバリデート又はインバリデートを発行する。次いで、プロセッサ・バスを経由してリプライやＳＡＣＫか送られて来るのを待つ。次いで、必要に応じて「ＣＲＡＢ」をスレーブのプロセッサに送り、システム・バスを経由してＲＢＵやＲＢＳのリプライをマスタのシステム・コントローラに送る。

【００５１】或る場合には、プロセッサ・バスにインバリデートを発行し、プロセッサ・バス経由でＳＡＣＫか送られて来るのを待ち、システム・バス経由でＰＳＡＣＫのリプライをマスタのシステム・コントローラに送る。

【００５２】或る場合には、システム・コントローラは、メモリに対するリード処理をキャンセルする。メモリ・リードをキャンセルしない場合には、システム・コントローラは、メモリ・リードが終了したこと，インバリデートを発行した場合はＳＡＣＫのリプライがあったことの両方の条件が満足した時に、システム・バスを経由してＲＢＵ／Ｓのリプライをマスタのシステム・コントローラに送る。

【００５３】マスタのシステム・コントローラにリプライを送った後、システム・コントローラは、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態を解除し、タグの更新を行う。

【００５４】図７はプロセッサ・バスからＷＲＢ要求（ライトバック要求）を受け取ったマスタのシステム・コントローラの動作を示す図である。システム・コントローラは、プロセッサ・バスからＷＲＢ要求を受け取ると、処理中のメモリ・アクセス要求のアドレスと今回のＷＲＢ要求のアドレスが一致しているか否かを調べる。両者が一致している場合は、今回のＷＲＢ要求をブロッキング・バッファに格納する。両者が一致していない場合は、今回のＷＲＢ要求をシステム・バスに出力する。

【００５５】ＷＲＢ要求をシステム・バスに出力した後、システム・コントローラはＷＲＢ要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。そして、システム・コントローラは、読み込んだキャッシュの状態を参照して種々の動作を行う。

【００５６】或る場合には、「ＰＢ　Ｓｒｅｐｌｙ　ＷＢＣＡＮ　」を発行する。即ち、プロセッサ・バスを経由して「ライト・バックをキャンセルせよ」と言うリプライをＷＲＢ発行元のプロセッサに送る。

【００５７】或る場合には、「ＰＢ　Ｓｒｅｐｌｙ　ＷＡＢ　」を発行する。即ち、プロセッサ・バスを経由して「ライトバックせよ」と言うリプライをＷＲＢ発行元のプロセッサに送る。次いで、ＷＲＢの対象となっているメモリ空間が自己のノードの中に存在するか，他のノードの中に存在するかを調べる。自己のノードの中に存在する場合は、自己のメモリにＷＲＢに付加されているデータを書き込む。他のノードの中に存在する場合には、「ＴＷ　Ｒｅｐｌｙ　ＳＷＢ　」を発行する。即ち、システム・バスを経由して「ライト・ブロックをメモリに書き込め」と言うリプライをスレーブのシステム・コントローラに送る。

【００５８】メモリ・ライトなどの必要な処理を行った後、システム・コントローラは「ＴＷ　ＣＥＮＤ」を発行する。「ＴＷ　ＣＥＮＤ」が発行されると、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態が解除され、次いでタグの更新が行われる。

【００５９】図８はシステム・バスからＷＲＢ要求を受け取ったスレーブのシステム・コントローラの動作を示す図である。システム・コントローラは、システム・バスからＷＲＢ要求を受信すると、ＷＲＢ要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。

【００６０】次いで、システム・コントローラは、受信したＷＲＢ要求で指定されたメモリ空間が他ノードの中に存在するか，自ノードの中に存在するかを調べる。自ノードの中に存在する場合には、マスタのシステム・コントローラから「ＴＷ　Ｒｅｐｌｙ　ＳＷＢ」が送られて来るのを待ち、送られて来たならばメモリ・ライトを行い、「ＴＷ　ＣＥＮＤ」を発行する。「ＴＷ　ＣＥＮＤ」が発行されると、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態は解除され、タグの更新が行われる。

【００６１】図９はプロセッサ・バスからＷＲＩ要求（ライト・インバリデート要求）を受け取ったマスタのシステム・コントローラの動作を示す図である。システム・コントローラは、プロセッサ・バスからＷＲＩ要求

を受け取ると、処理中のメモリ・アクセス要求のアドレスと今回のＷＲＩ要求のアドレスが一致しているか否かを調べる。両者が一致している場合は、今回のＷＲＩ要求をブロッキング・バッファに格納する。両者が一致していない場合は、今回のＷＲＩ要求をシステム・バスに出力する。

【００６２】ＷＲＩ要求をシステム・バスに出力した後、システム・コントローラはＷＲＩ要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。そして、システム・コントローラは、読み込んだキャッシュの状態を参照して種々の動作を行う。

【００６３】或る場合には、「ＰＢ　ＳＲＥＱ　ＩＮＶ」を発行する。即ち、プロセッサ・バスを経由して「インバリデートせよ」と言う要求をキャッシュの状態がＳ，Ｏ又はＭのプロセッサに送る。次いで、プロセッサ・バス経由でＳＡＣＫが送られて来るのを待つ。或る場合には、システム・バス経由でＳＡＣＫが送られて来るのを待つ。

【００６４】或る場合には、「ＰＢ　Ｓｒｅｐｌｙ　ＷＡＢ　」を発行する。次いで、ＷＲＩの対象となっているメモリ空間が自己のノードの中に存在するか，他のノードの中に存在するかを調べる。自己のノードの中に存在する場合は、自己のメモリにＷＲＩに付加されているデータを書き込む。他のノードの中に存在する場合には、「ＴＷ　Ｒｅｐｌｙ　ＳＷＢ　」を発行する。すなわち、システム・バスを経由して「ライト・ブロックをメモリに書き込め」と言うリプライをスレーブのシステム・コントローラに送る。この処理は、待ち合わせが終了した後で行われる。

【００６５】待ち合わせやメモリ・ライトが終了した後、システム・コントローラは「ＴＷＣＥＮＤ」を発行する。「ＴＷ　ＣＥＮＤ」が発行されると、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態が解除され、タグの更新が行われる。

【００６６】図１０はシステム・バスからＷＲＩ要求を受け取ったスレーブのシステム・コントローラの動作を示す図である。システム・コントローラは、システム・バスからＷＲＩ要求を受信すると、ＷＲＩ要求のアドレスによって定まるキャッシュの状態をスヌープ・バスに出力し、次いでスヌープ・バス上のキャッシュの状態を読み込む。そして、システム・コントローラは、読み込んだキャッシュの状態を参照して種々の動作を行う。

【００６７】或る場合には、プロセッサ・バスにインバリデートを発行し、プロセッサ・バス経由でＳＡＣＫが送られて来るのを待ち、送られて来たならばシステム・バス経由でＳＡＣＫをマスタのシステム・コントローラに送る。或る場合には、ＷＲＩ要求で指定されるメモリ空間が自己のノードに存在するか否かを調べ、自己のノードに存在する場合にはシステム・バス経由でＳＷＢが送られて来るのを待ち、送られて来たならばメモリ・ライトを行い、「ＴＷ　ＣＥＮＤ」を発行する。「ＴＷ　ＣＥＮＤ」が発行されると、ＣＥＮＤ待ちになっているブロッキング・バッファの要求の待ち状態は解除され、タグの更新が行われる。

【００６８】

【発明の効果】以上の説明から明らかなように、本発明によれば、キャッシュのコヒーレンシーを保つための，システム・コントローラ間のバス・トランザクションを少なく出来る。また、プロセッサのキャッシュに対する処理の開始も早くなるため、システム全体の性能の向上が期待できる。

【図面の簡単な説明】

【図１】本発明の原理説明図である。

【図２】本発明のマルチプロセッサの構成例を示す図である。

【図３】本発明のシステム・コントローラの構成例を示す図である。

【図４】本発明の動作を説明する図である。

【図５】プロセッサ・バスからリード要求を受け取ったマスタのシステム・コントローラの動作を示す図である。

【図６】システム・バスからリード要求を受け取ったスレーブのシステム・コントローラの動作を示す図である。

【図７】プロセッサ・バスからＷＲＢ要求を受け取ったマスタのシステム・コントローラの動作を示す図である。

【図８】システム・バスからＷＲＢ要求を受け取ったスレーブのシステム・コントローラの動作を示す図である。

【図９】プロセッサ・バスからＷＲＩ要求を受け取ったマスタのシステム・コントローラの動作を示す図である。

【図１０】システム・バスからＷＲＩ要求を受け取ったスレーブのシステム・コントローラの動作を示す図である。

【符号の説明】

１００　ノード
１１１　プロセッサ
１１２　プロセッサ
１２１　タグ部
１２２　タグ部
１３０　システム・コントローラ
１４０　クロスバ・スイッチ
１５１　メモリ
１５２　メモリ
１６０　バス・ブリッジ
２００　ノード
２１１　プロセッサ

15

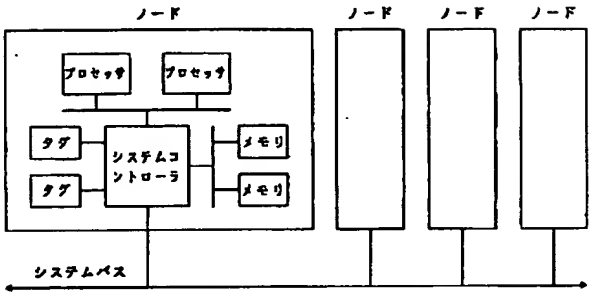
２１２　プロセッサ
２２１　タグ部
２２２　タグ部
２３０　システム・コントローラ
２４０　クロスバ・スイッチ
２５１　メモリ
２５２　メモリ
２６０　バス・ブリッジ

16

３００　ノード
３１１　プロセッサ
３１２　プロセッサ
３２１　タグ部
３２２　タグ部
３３０　システム・コントローラ
３４０　クロスバ・スイッチ
３６０　バス・ブリッジ

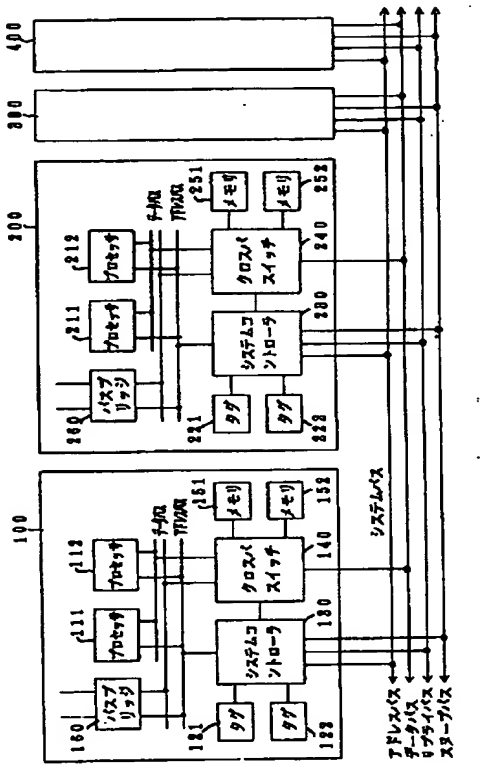
【図１】

本発明の原理説明図



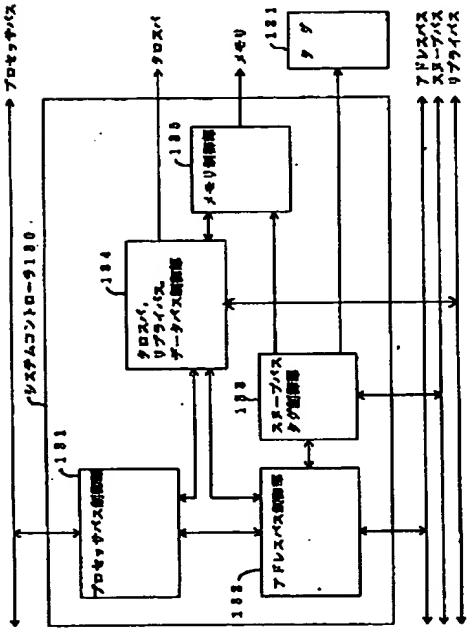
【図２】

本発明のマルチプロセッサの構成例



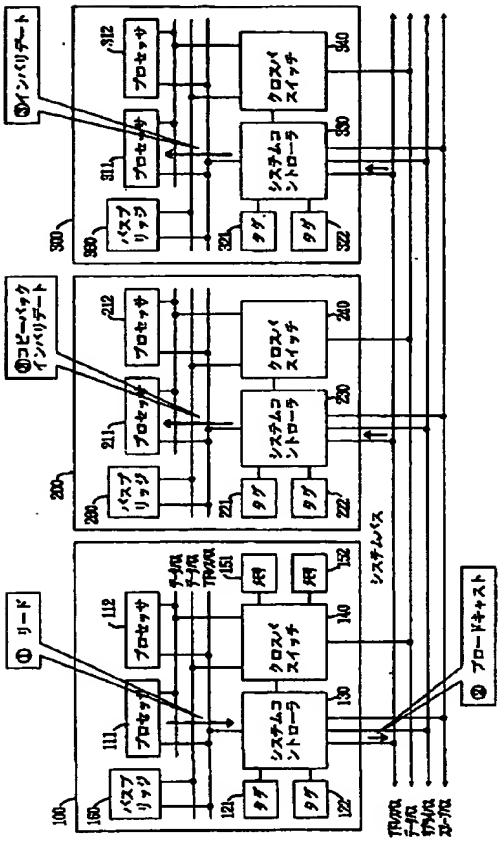
【図３】

システム・コントローラの構成例



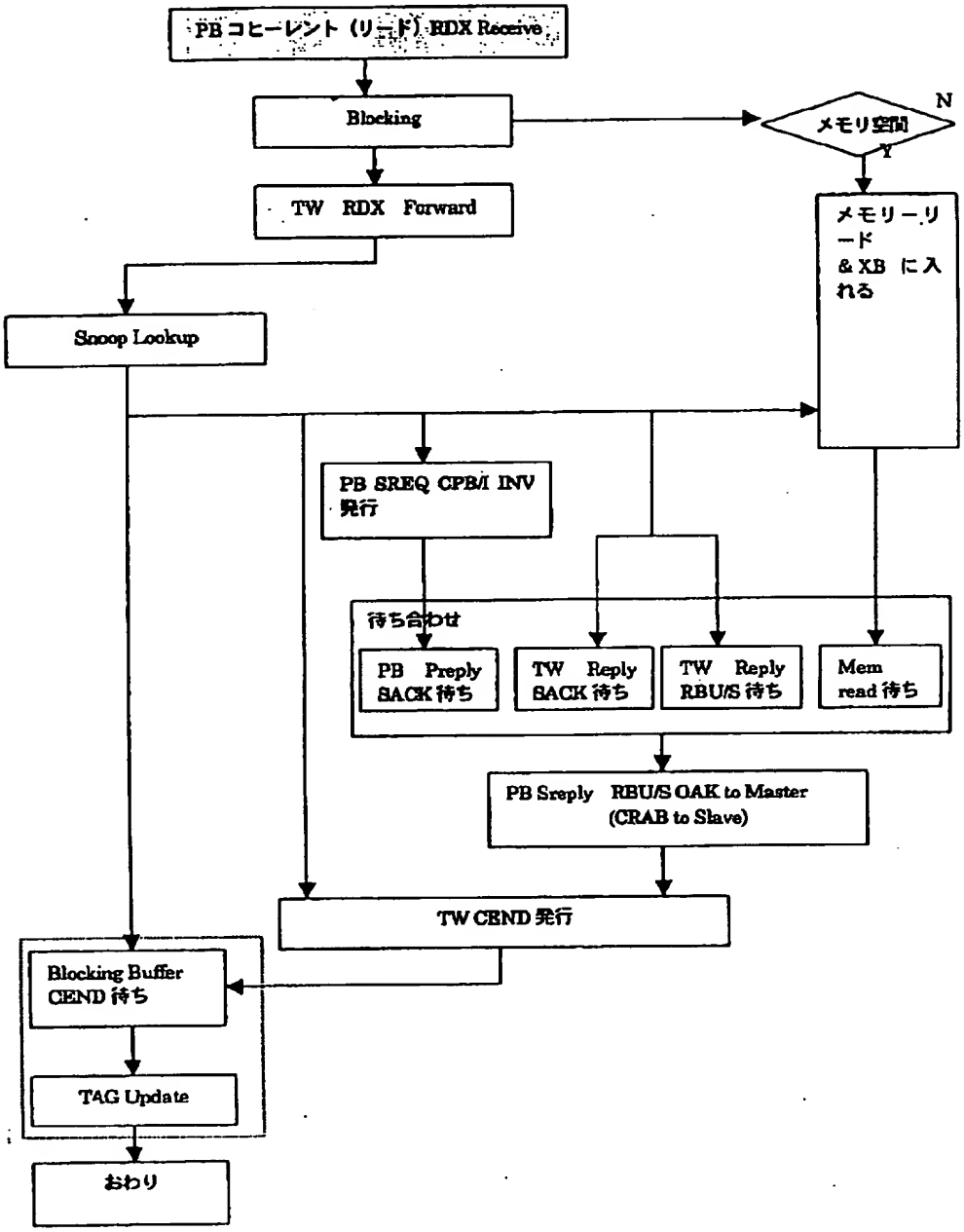
【図４】

本発明の動作を説明するための図



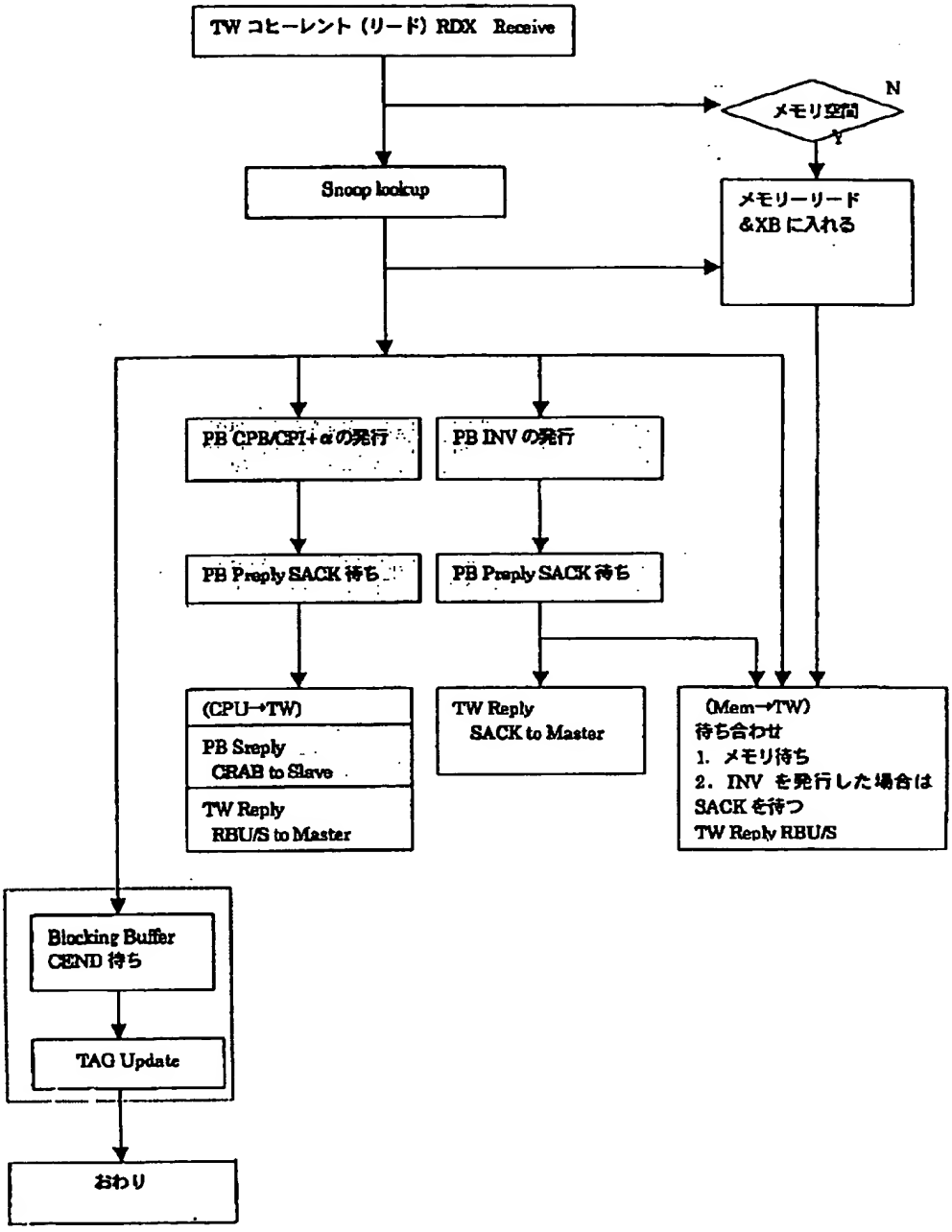
【図５】

プロセッサ・バスからリード要求を受け取ったマスタのシステム・コントローラの動作



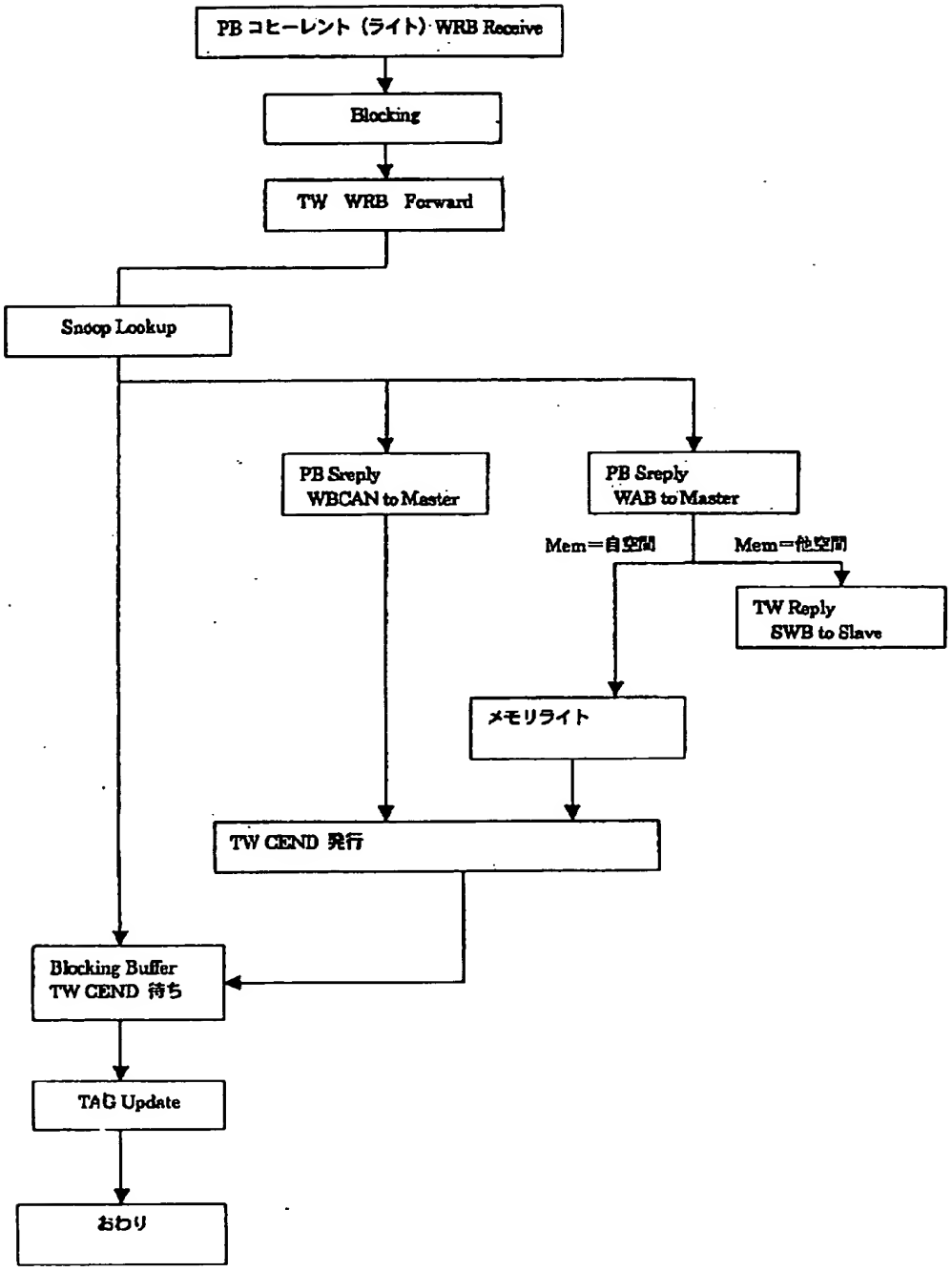
【図６】

システム・バスからリード要求を受け取ったスレーブのシステム・コントローラの動作



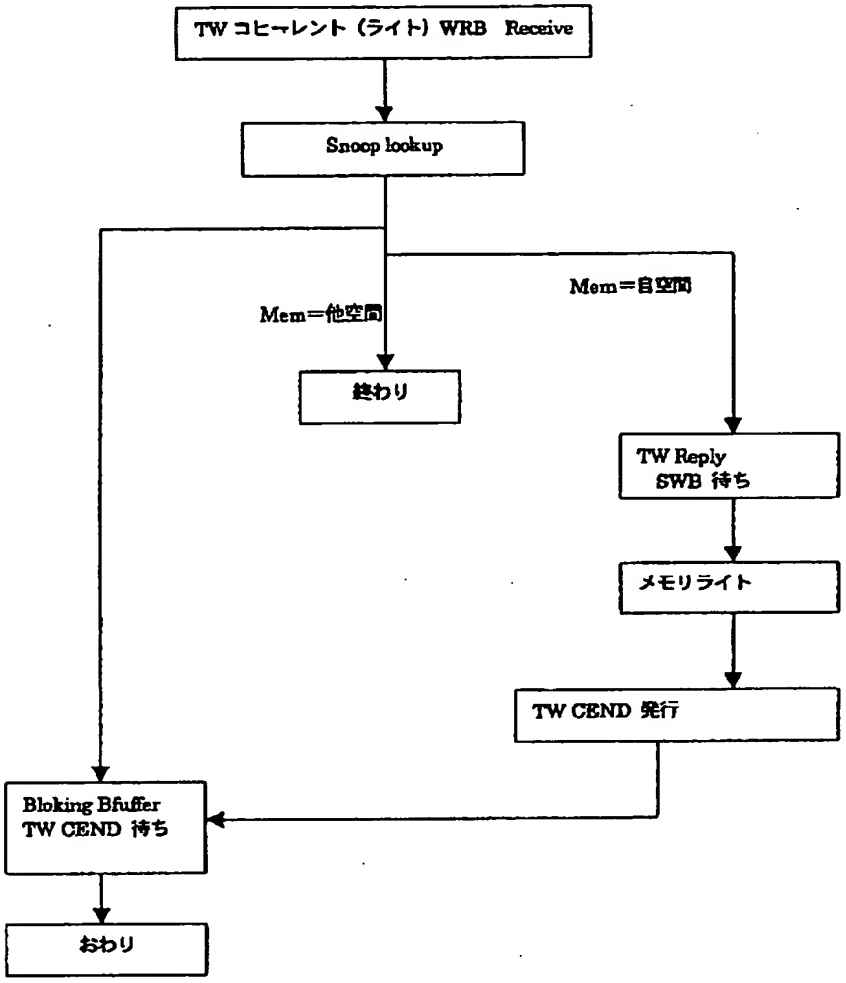
【図７】

プロセッサ・バスからＷＲＢ要求を受け取ったマスタのシステム・コントローラの動作



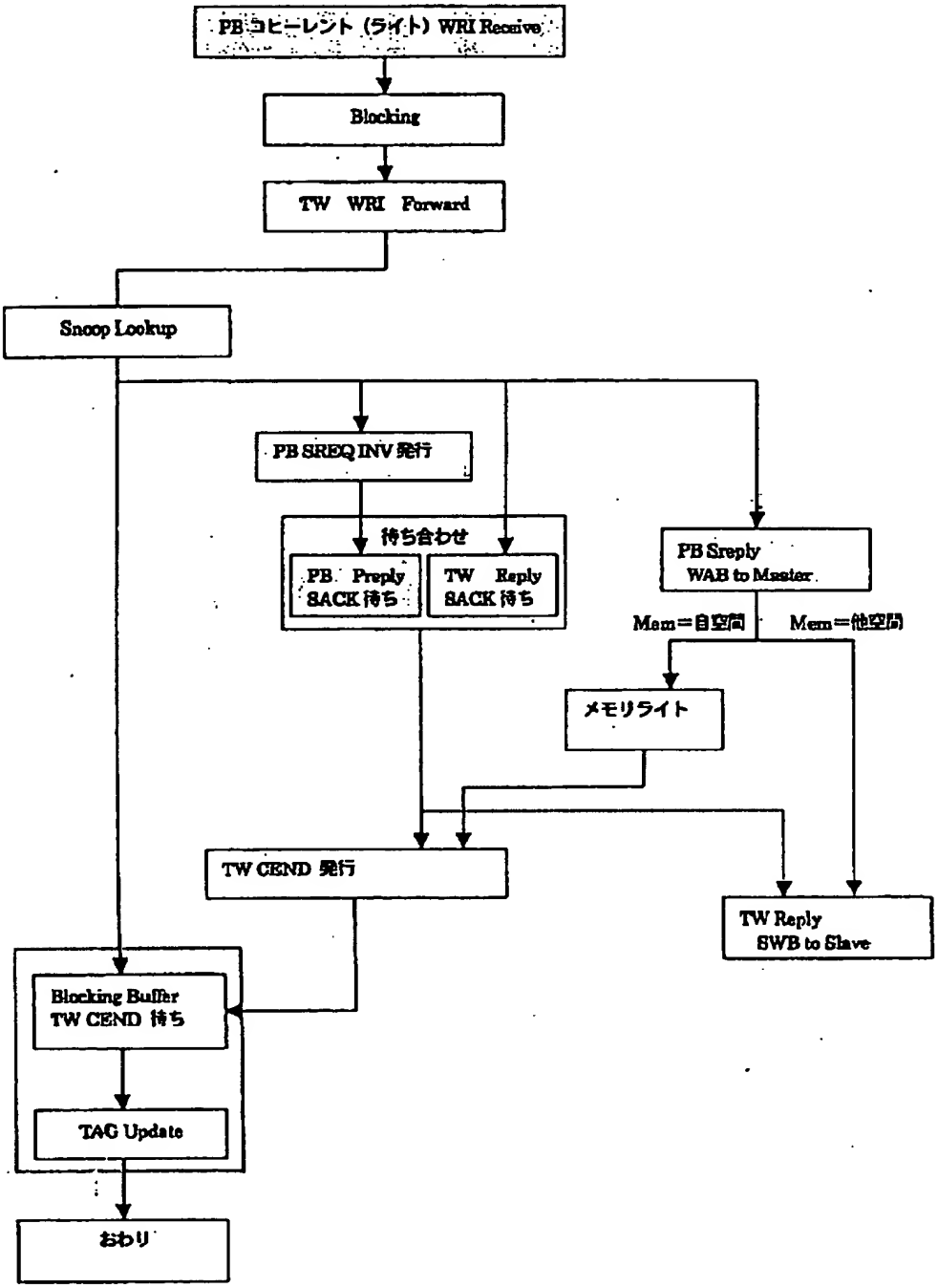
【図８】

システム・バスからＷＲＢ要求を受け取ったスレーブのシステム・コントローラの動作



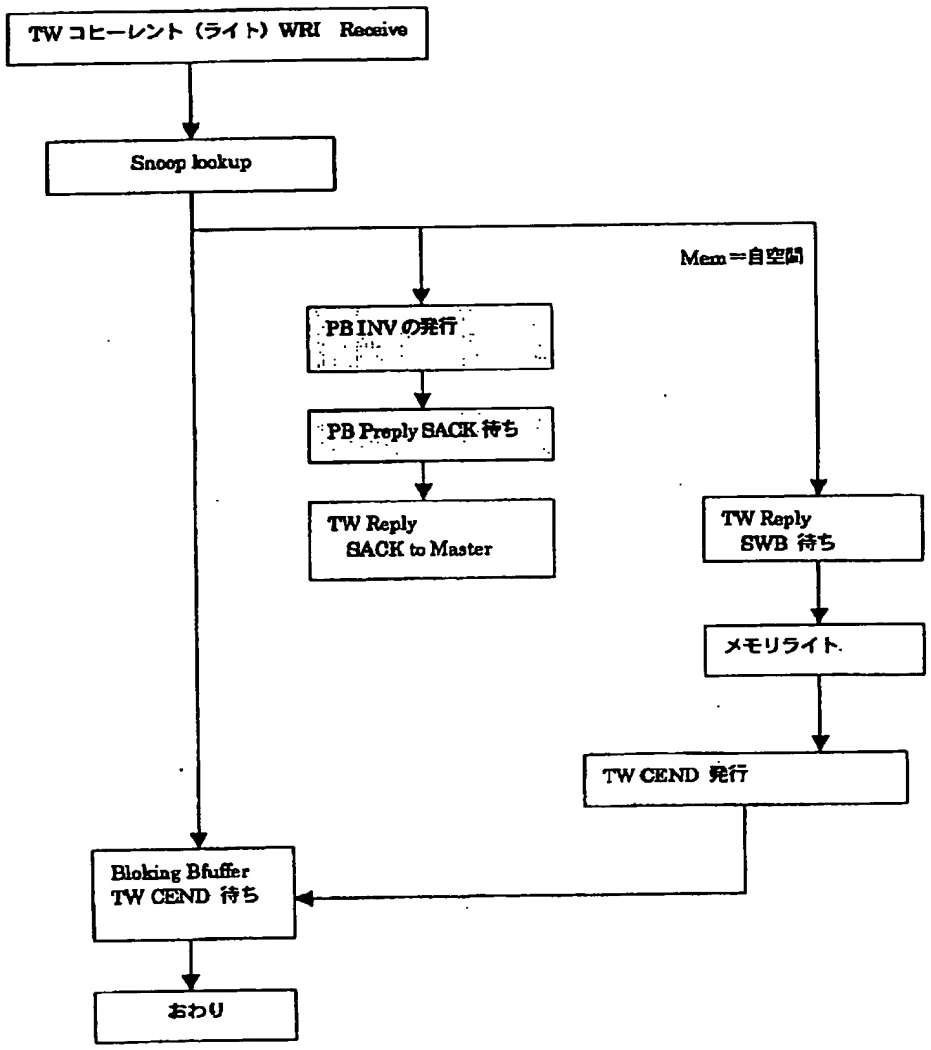
【図９】

プロセッサ・バスからＷＲＩ要求を受け取ったマスタのシステム・コントローラの動作



【図１０】

システム・バスからＷＲＩ要求を受け取ったスレーブのシステム・コントローラの動作



フロントページの続き

(72)発明者　佐藤　正美
石川県河北郡宇ノ気町字宇野気ヌ98番地の２　株式会社ピーエフユー内

Ｆターム(参考)　5B045 BB12 DD12 DD13